Metadata in NIEM 0.2

Georgia Tech Research Institute

December 20, 2005

Contents

1 Requirements

1. Provide a consistent mechanism to attach metadata properties (similar to but not limited to current XML attributes on SuperType) to any component of the model (object, association, role, or property) in any namespace.

2. Allow sets of metadata to be extended with additional properties for local requirements, without using wildcards.

3. Enable metadata properties to be repeated (cardinality `0` to `unbounded`)

4. Identify the types that a particular set of metadata properties can modify.

2 Description of Technique

This technique provides a general method for applying metadata and additional content to data objects in NIEM 0.2. It enables users to create a block of metadata and apply it to objects in exchanges. An object states what metadata applies to it using the `metadata` attribute.

In this example, we have a specific reported date for a person object:

```
<Person s:metadata="MD">
  <PersonName>
    <PersonGivenName>Adam</PersonGivenName>
    <PersonSurName>Brooks</PersonSurName>
  </PersonName>
  <PersonBirthDate>1960-10-07</PersonBirthDate>
</Person>

<Metadata s:id="MD">
  <ReportedDate>2005-08-01</ReportedDate>
</Metadata>
```

This example has a few interesting features:

- The person object refers to its metadata
  - The reference uses the attribute `s:metadata`
  - The reference is to the object with id `MD`
- The metadata is a separate element
  - The element is called `Metadata`.
  - The metadata object has the id ‘`MD`.
- The ID is conveyed with the attribute `s:id`
  - The metadata object contains an element `ReportedDate`

This is the core syntax. There are additional features that make this technique interesting.

2.1 Additional Cardinality of Metadata

This technique allows additional cardinality in metadata. Under GJXMD 3.0, each attribute may appear at most once. Under this method, the number of times a piece of information occurs may be controlled via the usual methods for elements in types.

```
<Metadata s:id="MD">
  <CommentText>Picked up on 12/20/02</CommentText>
  <CommentText>Released up on 12/22/02</CommentText>
  <CommentText>... additional comments ...</CommentText>
</Metadata>
```

2.2 Additional complexity of metadata

This technique allows for metadata information to be defined as is usual for elements. Elements may be of simple types, reference types, or structured types: This example has the reporting person as a reference to another person object:

```
<Metadata s:id="MD">
  <ReportingPersonRef s:ref="CD"/>
</Metadata>
```

The following example has a ReportingPersonName of a structured type:

```
<Metadata s:id="MD">
  <ReportingPersonName>
    <PersonGivenName>Charles</PersonGivenName>
    <PersonSurName>Davis</PersonSurName>
  </ReportingPersonName>
</Metadata>
```

2.3 Multiple blocks of metadata

This technique enables the application of multiple blocks of metadata. For example, a user may wish to apply super type metadata as well as custom metadata. The instance for this may look like the following:

```
<Person s:metadata="M1 M2">
  <PersonName>
    <PersonGivenName>Adam</PersonGivenName>
    <PersonSurName>Brooks</PersonSurName>
  </PersonName>
  <PersonBirthDate>1960-10-07</PersonBirthDate>
</Person>

<Metadata s:id="M1">
  <ReportedDate>2005-08-01</ReportedDate>
</Metadata>

<my:Metadata s:id="M2">
  <my:DatabaseID>2829019291</my:DatabaseID>
</my:Metadata>
```

This example shows two metadata blocks. Both are linked from the person object's `metadata` attribute. The first metadata block indicates the reported date. The second indicates a custom, extension database identifier.

This is made possible because of the way `s:metadata` is defined. The `metadata` attribute is of type `xsd:IDREFS`, which enables references to multiple targets. See [XMLSCHEMA-2], or [XML-INFOSET] for background on `IDREFS`.

2.4 Reuse of Metadata

This technique enables a block of metadata to be reused in multiple locations. A block of metadata may be defined once, and labeled (e.g. `Source1`, below). Then it may be reused by multiple objects. This creates a method similar to CSS classes, for identifying different types or sources of information.

In the following example, there are two explicit sources:

- Source 1 defines `PersonName`, `PrimaryContactInformation`, and `PersonBirthDate`
- Source 2 defines `Residence` and `Employment`.

```
<Person>
  <PersonName s:metadata="Source1">
    <PersonGivenName>Adam</PersonGivenName>
    <PersonSurName>Brooks</PersonSurName>
  </PersonName>
  <Residence s:metadata="Source2">
    ....
  </Residence>
  <PrimaryContactInformation s:metadata="Source1">
    ....
  </PrimaryContactInformation>
  <Employment s:metadata="Source2">
    ....
  </Employment>
  <PersonBirthDate s:metadata="Source1">
    1960-10-07
  </PersonBirthDate>
</Person>

<Metadata s:id="Source1">
  ... data specific to source 1 ...
</Metadata>

<Metadata s:id="Source2">
  ... data specific to source 2 ...
</Metadata>
```

2.5 Metadata Mechanism is Independent of NIEM Schema Release

This technique makes code table schemas, and additional schemas independent of the main NIEM schemas. A code schema will import the structures namespace, from which it will obtain the attribute `s:metadata`.

The following example shows a vehicle registration type code, which is of a type from NCIC.

```
<VehicleRegistration>
  <VehicleRegistrationPlateTypeCode s:metadata="PCMD">
    BU
  </VehicleRegistrationPlateTypeCode>
</VehicleRegistration>

<Metadata s:id="PCMD">
  ... metadata relevant to the plate code ...
</Metadata>
```

The schema definition for this would not need to involve a per-release proxy. Instead, all versions of the NCIC schemas could be derived from the same ‘`structures` schema, which provides the linking mechanism. The NIEM Schema would define the element, using the type from the NCIC schema:

```
<element name="VehicleRegistrationPlateTypeCode"
  type="NCIC:LITType"/>
```

The NCIC schema would create a simple type for the license plate code values:

```
<xsd:simpleType name="LITSimpleType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="AM"/>
    <xsd:enumeration value="AP"/>
      ...
    <xsd:enumeration value="VF"/>
    <xsd:enumeration value="ZZ"/>
  </xsd:restriction>
</xsd:simpleType>
```

The schema would then create a complex type. The complex type would be used as the type of elements. The type would be derived from a type in the structures namespace. This complex type would provide the `s:metadata` attribute.

```
<complexType name="LITType">
    <simpleContent>
    <restriction base="s:SimpleObjectType">
      <simpleType>
        <restriction base="this:LITSimpleType"/>
      </simpleType>
    </restriction>
  </simpleContent>
</complexType>
```

The definition of super type metadata in the main NIEM schema would indicate that it is applicable to all objects. This example applies to all *Objects* from the structures namespace.

```
<complexType name="SuperTypeMetadataType">
  <annotation><appinfo>
    <i:appliesTo i:name="Object"
        i:namespace="http://www.it.ojp.gov/structures/2.0"/>
  </appinfo></annotation>
  <complexContent>
    <extension base="s:MetadataType">
      <sequence>
        <element ref="j:CommentText"
            minOccurs="0" maxOccurs="unbounded"/>
        <element ref="j:CriminalInformationIndicator"
            minOccurs="0" maxOccurs="unbounded"/>
        ...
        <element ref="j:ReportedDate"
            minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

2.6 Metadata May be Defined to Apply to Specific Types of Objects

A metadata block may be defined to apply to a specific class of object. For example, a metadata block may apply to `PersonType` from the NIEM. This is expressed via appinfo in the schema:

```
<complexType name="MyPersonMetadataType">
  <annotation><appinfo>
    <i:appliesTo i:name="PersonType"
        i:namespace="http://niem.gov/niem/universal/0.2"/>
  </appinfo></annotation>
  <complexContent>
    <extension base="s:MetadataType">
      <sequence>
        <element ref="my:MyPersonID"
            minOccurs="0" maxOccurs="unbounded"/>
        ...
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Here we have a metadata block that defines the element `MyPersonID`, which may be applied to any `PersonType` object.

2.7 Metadata may be defined to apply to links between objects

A metadata block may be defined that applies to links between objects, not the objects themselves. Take as an example the special case of the name of a person:

- The person is held in custody
- The definition of the name comes from external records
- The assignment of the name to the person is based on an eyewitness

This presents three separate blocks of metadata:

- The person has one block of metadata, stating that the data was entered via booking
- The name has a block of metadata, stating when the data was validated, and the data source from which it was obtained
- The assignment of the name to the person has additional metadata, indicating the witness.
  - The metadata on the link is expressed with an attribute called `linkMetadata`.

```
<Person s:metatadata="PM">
  <PersonName s:metadata="PNM" s:linkMetadata="LM">
    <PersonPrefixName>Mr.</PersonPrefixName>
    <PersonGivenName>Xavier</PersonGivenName>
    <PersonMiddleName>Laughton</PersonMiddleName>
    <PersonSurName>McAlester</PersonSurName>
    <PersonSuffixName>III</PersonSuffixName>
    <PersonFullName>
      Mr. Xavier Laughton McAlester, III
    </PersonFullName>
    <PersonNameInitialsText>XLM</PersonNameInitialsText>
  </PersonName>
</Person>

<Metadata s:id="LM">
  <!-- data specific to the link between
     the person and the person name -->
  <CommentText>Reported by witness</CommentText>
  <ReportingPersonName>
    <PersonGivenName>Edward</PersonGivenName>
    <PersonSurName>Fritz</PersonSurName>
  </ReportingPersonName>
  <ReportedDate>2002-10-01</ReportedDate>
</Metadata>

<Metadata s:id="PM">
  ... data specific to the person ...
</Metadata>

<Metadata s:id="PNM">
  ... data specific to the person name ...
</Metadata>
```

The attribute `linkMetadata` conveys information that can't be conveyed by the `metadata` attribute. It tells applications that the metadata does not apply to either object. Instead, it applies to the connection between the two objects.

3 Advantages

1. Enables application of metadata to objects, roles, associations, and simple properties.
2. Provides uniform syntax for application of metadata.
3. Provides strong XML Schema validation of the contents of a metadata block.
4. Provides advanced capabilities:
   (a) Enables multiple occurrences of a field in a metadata block.
   (b) Enables sophisticated metadata definition, including complex structures and references.
   (c) Enables data to have multiple metadata blocks. Each block may be defined separately.
   (d) Enables a block of metadata to be reused across many data items.
   (e) Enables metadata to be added to a type without explicit extension of that type.
5. Makes external schemas independent of SuperTypeMetadata, as was used in GJXDM 3.0. Also makes external schemas independent of specific NIEM release schemas. Removes the need for importing code schemas through proxies using circular import statements, as was done in GJXDM 3.0.

4 Disadvantages

1. Uses elements instead of attributes as the primary representation for metadata. Attributes may still be used (e.g. for units on a measurement), but SuperType-style metadata would be represented by elements
2. Does not provide for XML Schema validation of the applicability of a block of metadata to a data. This check must be made by non-XSD means.
3. Adds additional weight to appinfo for validation of metadata against data.
4. Adds additional constructs in the NIEM schemas and conformant schemas. These constructs must be specified in the Naming and Design Rules.
5. Requires that an application dereference an IDREF to obtain metadata on an object.

References

[XML-INFOSET] XML Information Set (Second Edition), W3C Recommendation 4 February 2004, Available at `http://www.w3.org/TR/2004/REC-xml-infoset-20040204/`

IDREFS at `#infoitem.attribute`

[XMLSCHEMA-2] XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 28 October 2004. Available at `http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/`

IDREFS at `#IDREFS`